

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-185507

(13) 公開日 平成9年(1997)7月15日

(51) Int. Cl. ⁵	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 9/38	3 5 0		G 0 6 F 9/38	3 5 0 A
9/45			9/44	3 2 2 F

特許請求 未請求 請求項の数 4 O L (全 6 頁)

(21) 出願番号 特願平7-342210

(22) 出願日 平成7年(1995)12月28日

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 西山 博

神奈川県川崎市麻生区王禅寺1089番地 株

式会社日立製作所システム開発研究所内

(74) 代理人 弁理士 小川 勝男

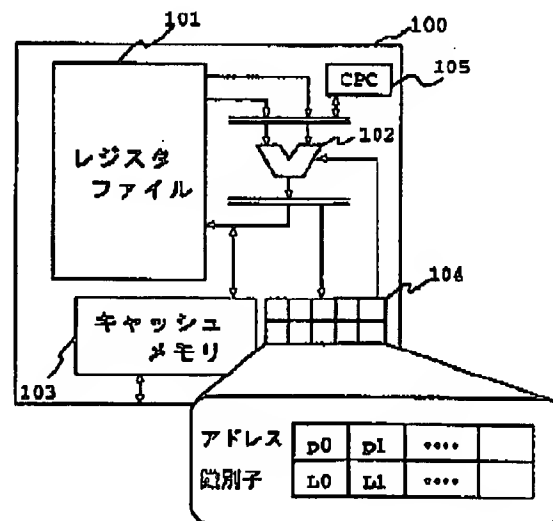
(54) 【発明の名称】 プロセッサ及びその制御方法

(57) 【要約】

【課題】 マイクロプロセッサにおいて、エイリアス関係にあるかどうかコンパイル時に不明なメモリ参照命令を、性能を低下することなく自由にスケジュールすることを可能とする。

【解決手段】 参照したメモリアドレスとその命令の識別子を記憶する表をハードウェア上に設け、メモリ参照を行なった際に、参照したアドレスと当該命令の識別子をこの表上に記憶する命令により、参照したアドレスと当該命令の識別子を登録する。後続して実行されるメモリ参照命令の参照アドレスが登録されたアドレスと同一であれば、表上に記憶された識別子の表す命令に対する補正コードを実行する。これにより、エイリアス関係のない場合に実行される命令数を増加することなく、メモリ参照を行なう命令を自由にスケジュールできるようになる。

図 1



(2)

特開平9-185507

1

【特許請求の範囲】

【請求項1】参照したメモリのアドレスとそのメモリを参照したメモリ参照命令の識別子の対応表を持ち、メモリからのデータのロードまたはストア時に参照対象のメモリのアドレスと当該メモリ参照命令の識別子を登録し、後続して実行されるメモリ参照命令の実行時にこの表を検査し参照対象アドレスが先に登録されたアドレスと同じであれば、登録された識別子に対応したメモリ参照命令の補正コードを実行して命令実行の補正を行なうことを特徴とするプロセッサ。

【請求項2】請求項1の参照対象のメモリアドレスと参照した命令の識別子の対応表に、参照対象のメモリアドレスと参照した命令の識別子の組を登録する機能を持つメモリ参照命令を備えることを特徴とするプロセッサ。

【請求項3】請求項1の参照対象のメモリアドレスと参照した命令の識別子の対応表を検査して、参照対象アドレスと同じアドレスが登録されていれば、対応表に登録された識別子に対応したメモリ参照命令に対する補正コードへ制御を移す機能を持つメモリ参照命令を備えることを特徴とするプロセッサ。

【請求項4】プログラミング言語のコンパイラにおいて、メモリ参照命令AとAに依存する可能性のあるメモリ参照命令Bに対して、命令Bを命令Aの上に移動し、請求項2および請求項3の命令を用いてメモリ参照命令Aおよびメモリ参照命令Bの間に依存がある場合に補正を行なうためのコードを生成することを特徴とするプロセッサ及びその制御方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明はマイクロプロセッサの命令実行方式に関し、さらに詳しくは、互いに依存関係を持つ可能性のあるメモリ参照命令をスケジューリングしてプログラムを高速に実行するために有効な命令実行方式とコンパイル方式に関する。

【0002】

【従来の技術】最近のマイクロプロセッサシステムでは、スーパースカラ、VLIWといった方式により命令レベルの並列性を利用し、プログラムを高速に実行できるようになっている。このような命令レベルの並列性を最大限に利用するためには、命令間の依存関係を正確に検出する必要がある。このような命令間の依存には、データ依存と制御依存がある。データ依存はさらに、フロー依存、逆依存、出力依存に分類される。コンパイラによってプログラムをコンパイルする際に、このようなデータ依存を常に正確に検出可能であるとは限らない。例えば、メモリ参照先のアドレスはプログラムのコンパイル時には不明な場合があるので、参照するメモリ領域が重なるかどうかをコンパイラが正確に判定することはできない。このような参照アドレスの重なりはエイリアスと呼ばれ、この関係が不明な場合には依存関係が存在する

2

ものと仮定してプログラムの最適化が行なわれる。特にC言語で記述されたプログラムではポインタが多用されるため、メモリ参照のエイリアス関係をコンパイル時に正確に検出できない場合が多い。プログラムの最適化時には、命令間に依存関係がない方がプログラムを並列化し性能を向上することができるため、エイリアス関係の不明なメモリ参照による依存によってプログラムの実行性能が低く抑えられてしまう。

【0003】このような場合に対処するため、例えば「A. Nicolau, Run-Time Disambiguation: Coping with Statically Unpredictable Dependencies, IEEE Transactions on Computers, Vol. 38, No. 5, pp. 663-678, 1989」や「A. S. Huang他, Speculative Disambiguation: A Compilation Technique for Dynamic Memory Disambiguation, Proceedings of the 21st Annual International Symposium on Computer Architecture, pp. 200-210, 1994」に述べられているように、実行時に動的にメモリ参照間の依存関係を調べる方式が研究されている。

【0004】これら従来の方式では、プログラムをコンパイルする際に、エイリアス関係が不明なことによって依存関係を持つかどうか不明なメモリ参照命令に関して、その参照対象アドレス同士を比較し、これらメモリ参照対象のアドレスが同一の場合と同一でない場合で異なるコードを選択して実行するように変換している。プログラムの実行時には、アドレスが同一かどうかに応じてそのどちらかが選択的に実行されることとなる。

【0005】

【発明が解決しようとする課題】上記従来の方式では、依存関係を持つ可能性のあるメモリ参照毎に、アドレスの比較命令と依存の有無によって実行される命令を生成する。この結果、実際にはメモリ参照を行なう命令間に依存がない場合にも実行されるコード量が増加し、却って性能が低下してしまう可能性が高くなる。

【0006】本発明の目的は、エイリアス関係にない場合に実行されるコード量の増加を低く抑えて実行時にメモリ参照命令間の依存関係を検出し、自由に命令スケジューリングを行なうための命令実行方式を提供することにある。

【0007】

【課題を解決するための手段】以下、説明のため、メモリ参照命令I0と後続するメモリ参照命令I1に対して、プログラムの実行速度向上のためには命令I1が命令I0より先に実行されるよう命令スケジューリングが必要があるが、参照アドレスが重なるかどうか不明のためにフロー依存または出力依存が存在する可能性があるものとする。上記課題を解決するため、本発明では以下のような手段を用いる。

【0008】(1)メモリ参照命令によって参照されたアドレスとその識別子を登録する表をハードウェア上に用意する。

(9)

特開平9-185507

3

【0009】(2) 他のメモリ参照命令Xの参照対象とエイリアス関係にある可能性のあるメモリ参照を行なう命令Yを先行的に実行する場合、(1)の表に参照アドレスとその識別子を登録する。

【0010】(3) メモリ参照命令Xの実行時に(1)の表を参照し、エイリアス関係にあるメモリ参照Yが先行的に実行されているならば、メモリ参照命令Yは誤った結果を参照しているので、その結果を補正するための命令を実行する。

【0011】具体的な命令実行は以下のように行なわれる。

【0012】(1) メモリ参照命令I1によって参照したアドレスAと命令I1の識別子の組を、ハードウェア上に用意された表に記録する。この時、アドレスAが既に表上に登録されているならば、そのエントリを書き換える。

【0013】(2) メモリ参照命令I0の実行時、I0が参照するアドレスBと(1)の表に登録されたアドレスを比較し、一致するものがあれば、メモリ参照命令の実行を中断して、表上に記憶されたアドレスAに対応する識別子に対応する補正コードへ制御を移す。一致するアドレス

【0014】(3) (2)で補正コードへ制御を移した場合、命令I0の識別子を利用して命令I0に対する命令実行の補正処理を行ない、命令I0の次の命令に制御を移す。

【0015】上記方法によれば、コンパイル時にエイリアス関係を持つかどうかを決定することができないメモリ参照命令に関して、実行時にエイリアス関係がなければ、実行される命令数を増加することなく自由に命令をスケジューリングすることが可能になり、プログラムの実行性能を向上することができる。

【0016】

【発明の実施の形態】以下、図面を参照しながら本発明の一実施例について説明する。本実施例では、RISC型の単一命令発行のマイクロプロセッサを仮定する。ただし、本発明はRISC型単一命令発行のプロセッサのみに適用対象を限定するものではなく、スーパースカラおよびVLIWといったプロセッサへも適用することが可能である。

【0017】図1は本発明を適用したマイクロプロセッサの例である。プロセッサ100において、101はレジスタファイル、102は演算器、103はキャッシュメモリ、104は本発明で特徴的な部分である参照したメモリアドレスとその命令の識別子の組からなる表(以下、エイリアス検出表と呼ぶ)である。また、105はエイリアス検出時に実行されていた命令のアドレスを記憶する特殊レジスタCPCである。本実施例では、この表に登録する命令の識別子としてエイリアス発生時に実行する補正コードのアドレスを用いる。

【0018】エイリアス検出表104を用いてエイリアスの検出と補正処理を行なうために、図2に示すように、LOAD命令およびSTORE命令それぞれに対して、エイリアス検

4

出表に参照アドレスとエイリアス発生時に実行補正を行なうためのコードの分岐先を登録する機能を持つSLOAD命令201およびSSTORE命令202、エイリアス検出表を検査し参照するアドレスが既に登録されている場合に、エイリアス検出表に登録された補正コードのアドレスに制御を移す命令であるCLOAD命令203およびCSTORE命令204を設ける。この図の各命令において、targetおよびsourceはそれぞれターゲットレジスタおよびソースレジスタ、addressは参照対象アドレス、adjustは補正コードアドレスを示している。

【0019】図3にSLOAD命令およびSSTORE命令の実行時の動作を示す。

【0020】まず、処理301で実行を開始する。処理302において参照対象のアドレスがエイリアス検出表に登録されているかどうかを検査する。既に登録されているならば、処理303で登録されているエントリの補正コードアドレスを更新する。登録されていなければ、処理304で新たなエントリを作成し、参照アドレスと誤って実行されたメモリ参照を補正するためのコードのアドレスを登録する。次に処理305で通常のLOAD、STORE処理を実行し、処理306で終了する。

【0021】図4にCLOAD命令およびCSTORE命令の実行時の動作を示す。まず、処理402で参照対象のアドレスがエイリアス検出表に登録されているかどうかを検査する。アドレスが登録されている場合、処理403でエイリアス発生時の命令アドレスを保存するためのレジスタCPCに現在実行中の命令のアドレスを格納し、エイリアス表に登録されている補正コードアドレスに分岐する。参照アドレスがエイリアス表に登録されていない場合、処理404で通常のLOAD、STORE処理を実行して、処理405で処理を終了する。

【0022】なお、図3および図4に示した命令の動作フローチャートは命令によって実行される処理の実現の1つの例であり、処理の並列実行等に制限を加えるものではない。

【0023】次に、これらの命令の具体的な適用例について示す。まず、以下に示すようなフロー依存の場合の例を考える。ここで、コンパイル時にp0とp2の指すアドレスとp1とp2の指すアドレスがエイリアス関係にあるかどうかは不明とする。このため、一般的には、命令12は命令10および11にフロー依存するものとして扱わなければならない。

【0024】

...

10: STORE *p0 ← r0

...

11: STORE *p1 ← r1

...

12: LOAD r2 ← *p2

...

50

(4)

特開平9-185507

5

命令I2を命令I0の上に移動する場合、命令I0およびI1をCSTORE命令に変換し、命令I2をSLOAD命令に変換する。SLOAD命令の最後のオペランドに指定したアドレスには、アドレスがエイリアス関係にあった場合の誤った命令実行を補正するためのコードとして、まず、レジスタCPCの内容に従ってエイリアスを生じた各命令に対する補正コードへ分岐する。次に、ストア命令I0に対する補正コードでは、まず命令I0のターゲットアドレスp0へソースレジスタr0の値をストアし、続いて命令I0のソースレジスタを命令I2のターゲットレジスタへコピーするコード、命令スケジューリングにより命令I0より上に移動された、命令I2のターゲットレジスタの値に依存する命令をコピーし、最後に命令I1の次の命令への分岐命令を生成する。同様に、ストア命令I1に対する補正コードでは、まず命令I1のターゲットアドレスp1へソースレジスタr1の値をストアし、続いて命令I1のソースレジスタを命令I2のターゲットレジスタへコピーするコード、命令スケジューリングにより命令I0より上に移動された、命令I2のターゲットレジスタの値に依存する命令をコピーし、最後に命令I1の次の命令への分岐命令を生成する。

【0025】以上の補正コード生成処理では、メモリへのストア命令や例外を引き起こす可能性のある命令など、システムに対して不可逆な操作を行なう命令に関しては命令I0およびI1より上に移動することを制限する。上の例を変換した結果は、以下に示すようになる。

```

【0026】
...
I2: SLOAD r2 ← *p2, L2
...
I0: CSTORE *p0 ← r0
L0n:
...
I1: CSTORE *p1 ← r1
L1n:
...
L2:
GOTO L1c if(CPC == I1)
L0c:
STORE *p0 ← r0
COPY r2 ← r0
...
GOTO L0n
L1c:
STORE *p1 ← r1
COPY r2 ← r1
...
GOTO L1n
...

```

このコードを実行する場合、命令I2を実行した時点で、アドレスp2と補正コードアドレスL2をエイリアス検出表

6

に登録する。次に命令I0を実行する時点で、アドレスp0とアドレスp2が異なっていれば、エイリアスは生じていないので通常と同様にレジスタr0の値をアドレスp0へ格納する。これに対して、アドレスp0とp1が同じであれば、アドレスp0へのストア処理を中断して、レジスタCPCにI0のアドレスを格納し、補正コードL2に制御を移す。ここでCPCはI0のアドレスを指しているのも、命令I0に対する補正コードを実行する。ここで、命令I2で参照した値はアドレスp0およびp2の指しているメモリの古い値であるので、まずレジスタr0に格納された値をアドレスp0に書き込み、次に、命令I0のソースレジスタr0の値を命令I2のターゲットレジスタr2にコピーしてr2の値を使用する命令を再実行し、命令I0の次の命令に制御を移す。

【0027】同様に、命令I1を実行した時点で命令I2との間にエイリアスが検出された場合、ストア処理を中断して命令I1に対する補正処理を実行する。

【0028】次に、出力依存の場合の例を示す。フロー依存の場合と同様にp0とp1の指すアドレスがエイリアス関係にあるかどうかは不明とする。このため、一般的には、命令I1は命令I0に出力依存するものとして扱わなければならない。

```

【0029】
...
I0: STORE *p0 ← r0
...
I1: STORE *p1 ← r1
...

```

I1をI0より前に移動した場合、命令I0をCSTORE命令に変換し、命令I1をSSTORE命令に変換する。SLOAD命令の最後のオペランドに指定したアドレスには、*p1、*p2で指されるアドレスがエイリアス関係の場合に、命令I1による誤った値の参照の補正処理を行なうコードとして、I1の次の命令への分岐命令を生成する。

```

【0030】
...
I1: SSTORE *p1 ← r1, L1
...
I0: CSTORE *p0 ← r0
40 L:
...
L1:
goto L
...

```

このコードを実行する場合、命令I1を実行した時点で、*p1の内容のレジスタr1へのロードを行なうと共にアドレスp1と補正コードアドレスL1をエイリアス検出表に登録する。次に、命令I0を実行する時点で、アドレスp0とアドレスp1が異なっていれば、通常と同様にレジスタp0の指すアドレスへレジスタr0の値を格納する。これに対

(5)

特開平9-185507

7

8

して、アドレスp0とp1が同じであれば、補正コードI1に制御を移す。ここで、命令I0でストア対象となっている値は、命令I1のストア操作によって上書きされる値であるので、補正コードでは何も行わず、命令I0の次の命令に制御を移す。

【0031】以上、本実施例では、エイリアス検出表に、参照アドレスと補正コードアドレスを格納したが、命令フィールド長の制約などにより補正コードアドレスを命令識別子として利用できない場合には、レジスタ等によって規定された共通の補正コードに制御を移し、命令に指定された識別子によって補正処理を選択するような実現を行なうことも可能である。

【0032】

【発明の効果】本発明によれば、メモリ参照命令の参照

アドレス間にエイリアス関係のない場合には、実行される命令数を増加することなく依存関係を持つ可能性のあるメモリ参照命令を自由にスケジューリングすることが可能になる。これにより命令間の待ちを減少したり命令レベルの並列度を増加することが可能となり、計算機プログラムの実行の高速化に効果がある。

【図面の簡単な説明】

【図1】本発明を適用したプロセッサの例。

【図2】本発明を利用するためのメモリ参照命令の例。

【図3】SLOAD、SSTORE命令の動作フローチャート。

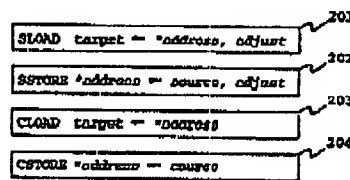
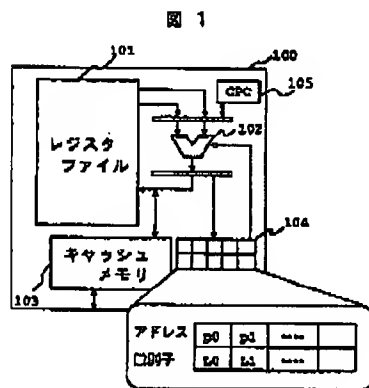
【図4】CLOAD、CSTORE命令の動作フローチャート。

【符号の説明】

101…レジスタファイル、102…演算器、103…キャッシュメモリ、104…エイリアス検出表。

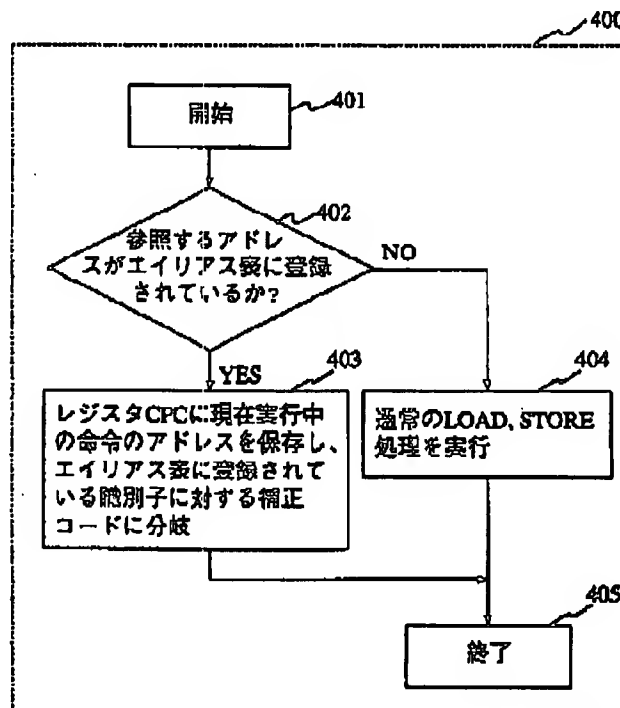
【図1】

【図2】



【図4】

図4



(6)

特開平9-185507

【図3】

図 3

